CoAP for Content-Centric Networks

Sridhar Srinivasa Subramanian, Joseph Pasquale, George C. Polyzos*
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
Email: sridhars@eng.ucsd.edu, pasquale@cs.ucsd.edu, polyzos@cs.ucsd.edu

***Abstract*—We analyze the design space for implementing the Constrained Application Protocol (CoAP) within Content-Centric Networks (CCN), identifying several CoAP specific scenarios and seeing how they map to CCNs. We present an evaluation, recommendations for implementations and extensions, and directions for future work. Our key result is that while several protocol features and flows map naturally, some communication patterns are more difficult to capture without modifications or additions to the CCN model. In addition to many conceptual and also non-performance related advantages, our experimental evaluation demonstrates that it is feasible and useful to implement CoAP over CCN, with the performance similar to that of CoAP over UDP over IP, and in several cases outperforming it significantly.**

***Index Terms*—Constrained Application Protocol, Content Centric Networking, Pull, Push, Group Communication**

I. Introduction

With the Internet of Things (IoT) gaining widespread adoption, it is estimated that there will be some 26 billion devices by the end of 2020 satisfying a wide range of applications [1]. Traditional Internet protocols may not be directly applicable to the constrained environment that IoT devices are expected to deal with, especially given the resource limitations of these devices. IoT networks often contain a large number of low-end, resource-constrained devices, typically equipped with limited computing power and required to operate over long time periods on battery power. Due to power constraints, IoT networks often employ low-energy Layer-2 technologies, such as IEEE 802.15.4, Bluetooth LE and low-power Wi-Fi, which usually operate with smaller MTU sizes and lower transmission rates compared to more traditional (e.g., Ethernet) links. To save energy, IoT nodes may not be always on as in wired networks. Moreover, an IoT system may be deployed in environments without wired network infrastructure (e.g., forests, underwater) and consequently has to rely on wireless mesh technologies to communicate [2]. To address these issues, several new protocols have been proposed for IoT environments. Two of the more popular messaging protocols are the Constrained Application Protocol (CoAP) and Message Queue Telemetry Transport (MQTT). CoAP is gaining prominence and popularity, and so our focus is on CoAP.

IoT needs to support heterogeneous and potentially mobile traffic with diverse application requirements [3]. IoT applications also inherently tend to be information centric (e.g., data consumers usually need data sensed from the environment without any reference to the subset of motes that will provide the requested information) [4]. This leads to the question of whether Content-Centric Networking (CCN) [5] is a natural fit for IoT environments. Prior work [6] on the feasibility and challenges of a CCN-based approach for the IoT has shown promising results. We argue that CCN, by closing the gap between application and network semantics, potentially offers a wide range of benefits in the IoT context. However, several CCN aspects need to be re-thought with constrained and low powered devices in mind. With CoAP receiving widespread acceptance, we believe it is worth exploring how to map this protocol to the CCN stack. Given that CoAP uses application-level URIs to identify resources, CoAP and CCN seem well matched.

The remainder of the paper is organized as follows: In Section II we briefly discuss the background of CoAP and CCN and then motivate the potential benefits that CCN could offer in the IoT context. In Section III we explore the design space around mapping CoAP to the CCN stack. In Section IV we discuss the implementation and evaluation in the Contiki Cooja environment. In Section V we discuss some additional issues and we present conclusions in Section VI.

II. Background and Motivation

CoAP [7] is a representational state transfer-style (RESTful) protocol developed for constrained devices that allows interactions between clients and servers over the Internet. In the environment of constrained devices, CoAP can be used for the manipulation of resources, such as sensor measurements or actuator states. Relying only on a simplified transport, CoAP uses UDP as the transport protocol by default. Since UDP does not implement end-to-end reliability CoAP provides optional application-layer end-to-end acknowledgements (ACKs). By setting the confirmable (CON) flag in an outgoing CoAP message, an end-to-end ACK is requested from the destination node. If no ACK is received within a retransmission timeout (RTO) interval, a retransmission is initiated for the outstanding transaction. Similar to HTTP, it supports requests with URIs, such as GET, PUT, POST, DELETE, etc., which can be used to fetch, manage and actuate IoT resources. CoAP also supports the Observe primitive in order to monitor resource state without having to poll every time.

CCN [5] is a communication architecture built on the concept of named data, where the identification and the transport of contents rely on their names and not on their

* Research performed while on leave from the Mobile Multimedia Laboratory, Department of Informatics, School of Information Sciences and Technology, Athens University of Economics and Business, Athens, Greece.

TABLE I. COMMUNICATION PATTERNS IN COAP

Communication type	Who is interested?	Who generates data?	Interest Validity	Data Validity	Cacheable
CoAP GET	CoAP client	CoAP server	Transient	x	Yes
CoAP PUT, POST, DELETE	CoAP Server	CoAP Client	Long-lived	Only once	No
Group communication	Nodes in group	Any node	Long-lived	Depends	Debatable
CoAP Observe	CoAP Client	CoAP server	Long-lived	Transient	Debatable

location. In contrast to today's IP-based, host-oriented, Internet architecture, CCN emphasizes content by making it directly addressable and routable. Endpoints communicate based on named data instead of IP addresses. CCN is characterized by the basic exchange of content request messages (called *interests*) and content return messages (called *content objects*, or simply data).

The benefits of adopting CCN for the IoT are numerous:

- *Support for efficient group communication* (multicast and in general multipoint communication): CCN naturally supports group communication.
- *Infrastructure-less communication*: CCN doesn't require an infrastructure, unlike an IP network. This can be especially valuable in emergency and disaster scenarios. Often, it is more economical as well.
- *Improved mobility support*: Mobility is easier to achieve in CCN as we are no longer routing via IP addresses, which might change in mobile scenarios.
- *Power savings*: Pervasive caching (and hence reduced bandwidth needs, efficient retransmissions, and potentially improved energy consumption for constrained devices).
- *Support for application layer pre-processing and data aggregation* (in intermediaries): Named Function Networking [8], which is an extension of CCN and conceptually similar to Active Networks is also generating a lot of interest.
- *Easy to build and configure applications*: Removing unnecessary middleware needs, in particular for applications and application protocols with a good match to CCN, such as CoAP that has a publish-subscribe profile.
- *Fine-grained security and access control*: CCN allows one to define more fine-grained name-based security and access control rules. It also secures the data as opposed to securing the channel(s).

III. COAP OVER CCN

In this section, we discuss how the primitives in CoAP can be mapped to CCN. The primitives available in CoAP include: CRUD primitives (GET, PUT, POST, DELETE), Group communication, Resource observe and discovery.

Each of the above is a fundamentally different form of communication when mapped onto CCN [9] (on-demand content distribution vs. publish/subscribe). Specifically, they differ in terms of who initiates the content flow (pull-based vs. push-based) and the validity of the content over various periods of time. For example, CoAP GET can be directly mapped to CCN by using the CoAP resource URIs as content names for which the consumers send interest packets on demand. The published content can also be cached opportunistically. Table I shows the differences in communication patterns of these primitives.

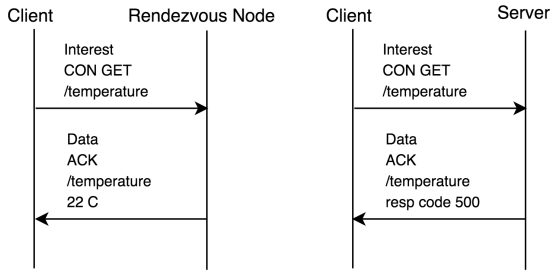

Fig. 1. Two GET requests with piggybacked responses. The first one is obtained from the rendezvous node's cache and the second one has an error that is returned in the response code.

In CCN, data transport is pull-based, i.e., a data consumer requests data via an interest packet. An interest can consume only one item of data (and vice-versa). However, in scenarios such as IoT actuation, group communication, etc., it is better to seek push-based solutions.

A. CoAP GET

The GET method retrieves a representation for the information that currently corresponds to the resource identified by the request URI [7]. The application level resource URI can be naturally mapped to a CCN content name. CoAP GET also follows the pull-based semantics as in CCN. An interest corresponds to a CoAP GET request and a response can be obtained as a data packet.

CoAP GET in the IP stack supports two forms of responses. One is a piggybacked response where the response is carried directly in the acknowledgement message. However, it may not be possible to return a piggybacked response in all cases. For example, a server might need more time to obtain the representation of the resource requested than it can wait to send back the acknowledgement message, without risking the client repeatedly retransmitting the request message. In a CCN realization of CoAP GET, the response can be obtained from the network if a cached copy is available in any of the neighbor nodes (on the reverse path). If the data is not already available, CoAP can use an interest message to trigger data generation and respond with a data packet with an appropriate response code. The flow is shown in Fig. 1.

B. CoAP POST, PUT and DELETE

The POST method requests that the representation enclosed in the request be processed. The PUT method requests that

the resource identified by the request URI be updated or created with the enclosed representation. The DELETE method requests that the resource identified by the request URI be deleted [7]. These requests are fundamentally different from CoAP GET requests as they require data to be pushed into the server, violating CCNs pull communication model. In order to support this, workarounds are necessary. This could be achieved by using long-lived interests, using interests to trigger interests, and interest overloading [10].

1) Long-lived interests: Long lived interests are discussed in [11]. Here, the provider of the resource sends a long-lived interest to indicate that it is interested in receiving the PUT, POST and DELETE messages for that particular resource. One way of supporting long-lived interests would be to issue interests that do not expire like regular interests, and consumers that would like to send PUT, POST and DELETE messages send notifications as a response for these interests. Since notifications do not erase the PIT entries, the resource provider can keep receiving the messages. [12]

One of the main advantages of long-lived interests is that, the PIT entries are already established in the network, and hence the content can be transferred with low latency. As for disadvantages, pending interests should be established at all the nodes from which packets are to be received. This means that we have to maintain a significant amount of state in the network. Another disadvantage is that it will be difficult for new network elements to discover outstanding interests, and there is a constant need to refresh the PIT entries.

2) Interest Trigger: Usage of interests to trigger another interest has been discussed in [11]. When a node wants to send a PUT/POST/DELETE request, it sends out an interest and the server responds back with a dummy data packet. Now that the server knows there is a node that is interested in sending a request, it sends another interest to fetch the PUT/POST/DELETE request. The flow is shown in Fig. 2.

An advantage of this method is that, unlike long-lived interests, with interest triggers there is no need to maintain persistent state in the network. Also because the payload is transmitted as data, it is easier to transmit secure content objects. The disadvantage is that this method leads to additional latency because of an additional interest-data exchange.

3) Interest Overloading: This method suggests augmenting the payload with an interest [11]. The rationale behind this method is that sending interests can be considered as a form of pushing data. In CCN, interests cannot carry payloads. However, there is no restriction on the size of a name. Leveraging this, in interest overloading, we suffix the interest name with the data and the server can acknowledge the interest by sending a data packet as an acknowledgement. (Recall that interest forwarding is based on name prefix, therefore the suffix has no impact on routing.)

The advantage, here again, is that there is no need to maintain persistent network state. Also, it can be supported without modifying CCN. In some ways, this allows us to mimic the IP style of pushing packets, while retaining the benefits of the CCN core for other types of messages. The disadvantage is that interest flooding is now more expensive (though this can be alleviated by good routing protocols).

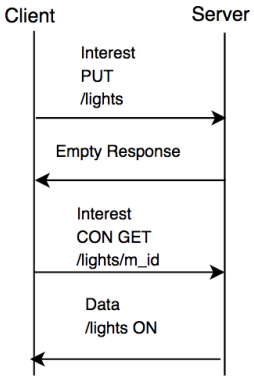

Fig. 2. Put request via interest trigger. An interest is used to convey the intent to send data.

Also, sending secure objects is now more difficult. Supporting payloads in interest packets could provide the much needed expressiveness.

C. Group Communication

Constrained devices can be large in number but are often related to each other in function or by location. For example, all the light switches in a building may belong to one group and all the thermostats may belong to another group. Groups may be pre-configured, e.g., before deployment or dynamically formed during operation. If information needs to be sent to or received from a group of devices, group communication primitives can reduce the latency of communication and the bandwidth requirements for a given application.

CoAP group communication is an optional extension proposed in RFC 7390 [13] that proposes an IP multicast-based approach to deliver messages to the group members. Multicast Protocol for Low power and Lossy Networks (MPL) is a current standard for multicast in low-powered lossy networks. Multicast in IP networks has in general not received widespread adoption. CoAP recommends use of resource directories to discover and broker multicast address allocation.

On the other hand, CCN, which is designed for content dissemination, has native and elegant support for group communication. The hierarchical name spaces can be leveraged creating fine-grained groups, which helps efficient prefix based routing. CCN simplifies the additional administrative burdens. For example, joining a group is as simple as sending an interest, and in many ways, completely removes the requirement of a resource directory. Group communication in CCN can be implemented by group members sending a long-lived interest and all messages could be sent as notifications. Additional research is needed in this direction as notifications which bypass PIT entries could cause congestion in the network.

D. CoAP Observe

To avoid additional overhead costs due to polling a resource, CoAP allows clients to observe resources, i.e., the server updates the client [14]. This can be supported in CCN using the

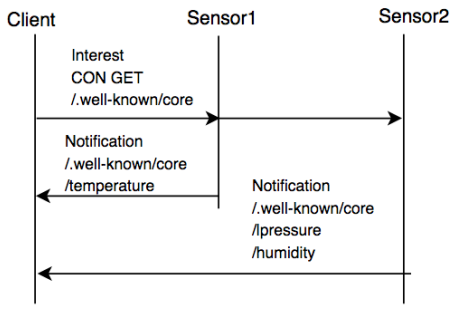

Fig. 3. Resource discovery by flooding. Nodes respond with their list of resources via notifications.

notification message in CCN to continuously send updates to the interested clients. This mechanism allows us to seamlessly extend it to group observation, where an entire group can observe specific resources. This can leverage the inherent group communication capabilities of CCN and significantly simplifies the application layer. For example, one interesting application would be all emergency alarms observing a specific sensor.

E. Resource Discovery

CoAP resource discovery enables clients to discover the resources a CoAP Server can provide [7]. This allows the client to determine what URIs are supported by the CoAP Server and the meaning of them. In IP networking, the request is to a specific server; in CCN, the request is to the network. Clients can discover resources by flooding interests, the servers respond via notifications so as to receive the data from multiple providers. We can also implement resource directories to minimize flooding. Clients can register their resources they want to make available to the resource directory. Resource discovery via flooding is shown in Fig. 3.

IV. Implementation and Evaluation

We added a CoAP layer on top of an existing open source CCNx implementation for the Contiki Operating System [15] using CCN as a layer 3 protocol. We have compiled and tested it for the MSP430 SkyMote architecture. CCN extensions such as long-lived interests and notifications were added. Table II shows the simulation configuration.

In the first experiment, we set up an equivalent CoAP over UDP and CoAP over CCN setup and measured multi-hop latencies in order to understand the overheads involved. Fig. 4 shows a very simple (so as to focus on latency vs. number of hops) multi-hop topology used for the experiments. Node 0 is the sink node, which sends CoAP GET requests and nodes 1, 2, 3 publish different resources and are 1, 2 and 3 hops away, respectively, from the sink node 0.

Fig. 5 shows the effect of the number of hops on the latency. The latencies of CoAP over CCN are slightly higher than the latency of CoAP over UDP. This could be attributed to the additional network load during interest flooding that causes link-level backoffs and the additional overhead in processing

TABLE II. Simulation Settings

Setting	Value
Wireless Channel	UDG Model with Distance Loss
Mote type	TMote Sky
MAC Layer	non-slotted CSMA + ContikiMAC
Radio Interface	CC2420 2.4 GHz (IEEE 802.15.4)
Network layers	IPv6/CCNx

Fig. 4. Topology for evaluating the effect of number of hops on the latency. Nodes 1, 2, 3 publish different resources and are 1, 2 and 3 hops away, respectively, from the sink node 0.

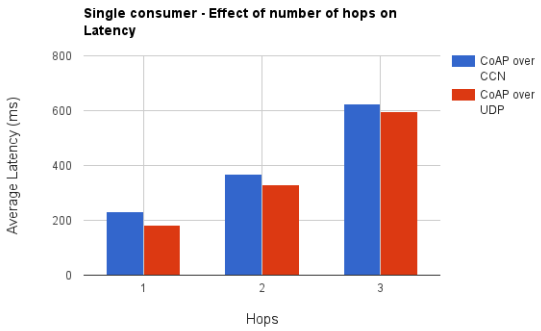

Fig. 5. Effect of number of hops on latency. Additional overheads in CCN add to the extra latency.

the data structures of CCN. Faster algorithms for CS lookup, PIT addition, etc., could improve this.

The second experiment focuses on the effect of caching on the network. We used basic Vanilla Interest Flooding (VIF) [6] with and without Content Store enabled and measured the number of interest and data packets sent in the network. The interest packets are broadcast and the data packets return in the reverse path. Duplicate interest packets are suppressed if a PIT entry for the interest already exists. In VIF without Content Store, the propagation of interests led to more nodes having a particular data item in the PIT entry, which led to more nodes obtaining the content to pass it on to the requester. This led to a major increase in the network load. This was avoided when caching was used for subsequent requests. Fig. 6 shows the effect of caching on network load.

In the third experiment, we investigated the effect of multiple consumers along the path for a single data item. Fig. 7 shows the topology with multiple consumers of the same data with a shared path from producer to consumer. With caching disabled, the network load does not increase linearly because when the number of consumers increases, duplicate interests are simply dropped by the upstream nodes. Caching had a

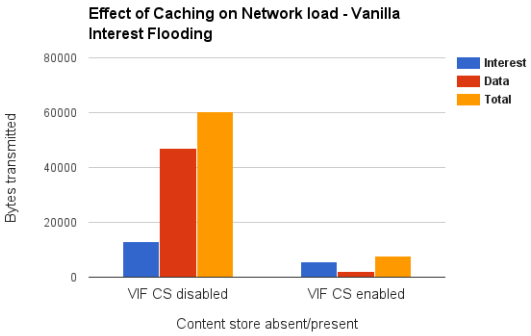

Fig. 6. Effect of caching on network load. Caching significantly reduces unnecessary propagation of interest and data packets during Vanilla Interest Flooding.

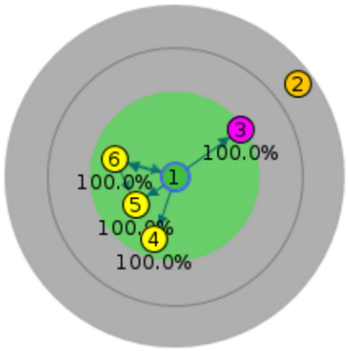

Fig. 7. Topology for evaluating the effect of number of consumers on path. Node 3 is the producer, Nodes 4, 5 and 6 are consumers and Node 1 is a network element on path.

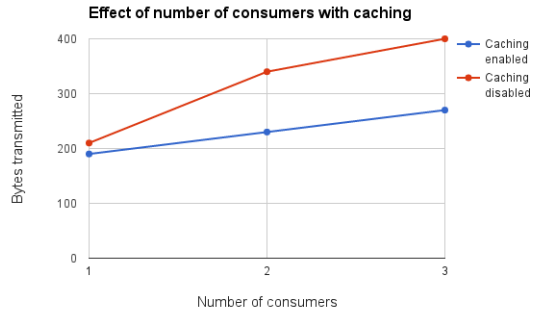

Fig. 8. Effect of number of consumers with caching. Caching significantly lowers network load with multiple consumers sharing common interests and data.

major impact on the network load as well as the latency. This is a very common use case where we have multiple actuators acting on the same command (content). Fig. 8 shows the results.

The fourth experiment was to verify the performance of Group Communication primitives provided by CoAP over UDP and CoAP over CCN. Fig. 9 shows the topology for

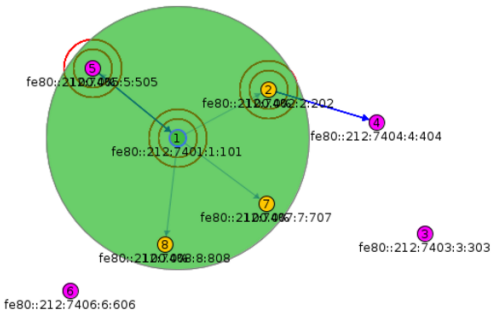

Fig. 9. Multicast group communication topology. Node 1 is the sender, nodes 3, 4, 5, and 6 are group members.

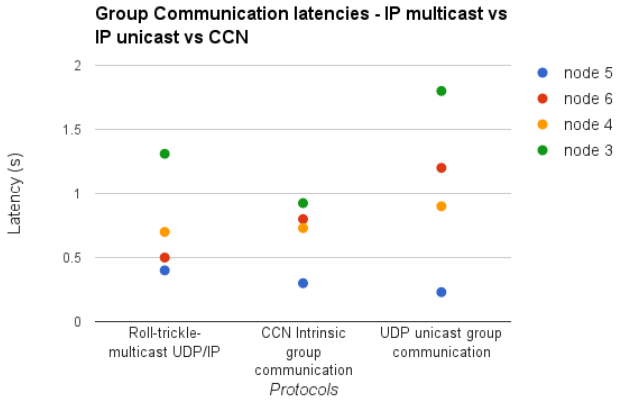

Fig. 10. Group communication latencies: UDP/IPv6 ROLL vs. UDP unicast vs. CCN. CCN's performance is comparable to IPv6 multicast.

Group Communication. Here, Node 1 is the root that intends to send data to the group members 3, 4, 5, and 6. In this experiment, the source node periodically sends a message to the group members and we measure the latency at each node. Fig. 10 shows the variation in the average latency of the nodes with the trickle-based ROLL multicast routing protocol [16], UDP unicast and CCN group communication. It can be seen that the CCN protocol has a better worst case latency, but apart from that they are fairly similar. UDP unicast has higher latencies as it does not benefit from the common path from source to destination because of increased number of messages.

V. Discussion

1) Impact on power savings: Unlike the IP model, in CCN the data can come from any node that has cached the data. An implication of this is that low-power devices do not necessarily have to be active for serving information. An infrastructure router or some other node on the fixed network could, for example, cache the data of the sensors in its range. Now, radio duty-cycling can be done on the low-powered device as per its needs. CoAP over IP recommends having middleboxes that act as proxies for saving power. The

CCN architecture can (intrinsically) avoid the need for this middlebox by pervasive caching in the network.

2) Security and Privacy: CoAP over UDP provides DTLS (Datagram Transport Layer Security) bindings for security [7]. This is not relevant for CCN as it provides content-based security, as opposed to IP's connection-based security. This allows the receiver to accept a data packet from anyone as long as its signature is valid and its content could be protected by encryption. These properties reduce the trust we must place on network intermediaries [5], but it also leads to new forms of attacks such as Content Poisoning and Interest Flooding. Privacy is another important challenge that needs to be addressed in CCN. 'Name' and 'Content' privacy are particularly relevant in the IoT context. While several counter-measures have been proposed [17], the feasibility and impact of those approaches should be studied more rigorously.

3) Impact of wireless networks: Most of the IoT devices typically use wireless low-power radios. Given the inherent broadcast nature of wireless communication, this is an effective way to disseminate interests, as a single interest packet can simultaneously cater to multiple network elements. With directional antennas and with additional metadata, efficient geo-casting is also possible. By overhearing content, nodes can opportunistically cache data when they are awake and be able to respond in highly mobile scenarios. By listening to other nodes, active neighbors can be discovered and use this information to minimize interest broadcasts. CCN optimized MAC layer protocols can be very useful and need further investigation.

4) Interest payloads: Supporting payloads in interests allows effectively overloading interests as push messages. This allows the support of an IP-style push communication while retaining the benefits of CCN. However, interest payloads can lead to congestion in the network when interests are flooded. To avoid this, congestion control mechanisms could be employed at the network layer.

5) Mobility and Long-lived interests: CCN can support mobility. However, long-lived interests and mobility could lead to stale PIT entries in the network. This could increase the load of the network unless invalidated. There should be mechanisms to detect the presence/absence of a node periodically and remove PIT entries that are no longer relevant. In CoAP over UDP, supporting mobility is much more difficult.

VI. Conclusion

CCN offers a clean solution to the problems faced by current IoT architectures. With CCN, the network can be better aware of the traffic characteristics (through the names of data) and adapt accordingly. It also gives the network self-organizing capabilities. However, CCN, in its current form, is not flexible enough to capture all the communication patterns, hence the need for an architecture that can support both publish/subscribe and on-demand content dissemination.

We discussed several tradeoffs in implementing CoAP over CCN. A key result is that CoAP can indeed be implemented in CCN with minor modifications. Its performance is also comparable with CoAP over IP, but with the potential to do significantly better in several scenarios. An important future direction is to investigate efficient routing protocols that take into account the dynamicity of the network to efficiently route to the content. Unnecessary overhead traffic and broadcasts that could be detrimental to the battery life of these devices should be avoided.

References

[1] P. Middleton, P. Kjeldsen, J. Tully, "Forecast: The Internet of Things,Worldwide, 2013," Gartner Inc., Tech. Rep. ID G00259115, Nov. 2013.

[2] W. Shang, Y. Yu, R. Droms, L. Zhang, "Challenges in IoT Networking via TCP/IP Architecture," NDN Project, Tech. Rep. NDN-0038, 2016.

[3] Y. Zhang, D. Raychadhuri, L. Grieco, R. Ravindran, G. Wang "ICN based Architecture for IoT - Requirements and Challenges," ICNRG Draft draft-zhang-icn-iot-architecture-00, Aug. 2015.

[4] M. Amadeo, C. Campolo, A. Iera, A. Molinaro, "Named data networking for IoT: an architectural perspective," Proc. IEEE European Conf. on Networks and Communications (EuCNC), Bologna, Italy, June 2014.

[5] V. Jacobson, D.K. Smetters,J.D. Thornton, M.F. Plass, N.H. Briggs, R.L. Braynard, "Networking Named Content," Proc. 5th ACM Intl. Conf. on Emerging Networking Experiments and Technologies (CoNEXT), Dec. 2009.

[6] E. Baccelli, C. Mehlis, O. Hahm, T.C. Schmidt, and M. Whlisch. "Information centric networking in the IoT: experiments with NDN in the wild," Proc. 1st ACM SIGCOMM Conference on Information-Centric Networking (ACM ICN), Paris, France, Sept. 2014.

[7] Z. Shelby, K. Hartke, C, Bormann, "The constrained application protocol (CoAP)," IETF RFC 7252, 2014.

[8] C. Tschudin and M. Sifalakis, "Named functions and cached computations," Proc. 11th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, Jan. 2014.

[9] A. Carzaniga, M. Papalini, A.L. Wolf, "Content-Based Publish/Subscribe Networking and Information-Centric Networking," Proc. ACM SIGCOMM Workshop on Information-Centric Networking, Aug. 2011.

[10] S.S. Seo, J.M. Kang, Y. Han, J.W.K. Hong, "Analysis and performance evaluation of data transport methods in Content-Centric Networking," Proc. 15th Asia-Pacific Network Ops. and Management Symp. (APNOMS), Sept. 2013.

[11] M. Amadeo, C. Campolo, A. Molinaro, "Internet of Things via Named Data Networking: The Support of Push Traffic," Proc. 5th Intl. Conference and Workshop on the Network of the Future, Paris, France, Dec. 2014.

[12] R. Ravindran and A. Chakraborti, "Support for Notifications in CCN," ICNRG Draft draft-ravi-ccn-notification-00, July 2015.

[13] A. Rahman and E. Dijk, "Group communication for the constrained application protocol (CoAP)," IETF RFC 7390, 2014.

[14] K. Hartke, "Observing resources in the constrained application protocol (CoAP)," IETF RFC 7641, 2015.

[15] B. Saadallah, A. Lahmadi, O. Festor, "CCNx for Contiki: implementation details," Tech. Rep. RT-0432, INRIA, 2012.

[16] J. Hui, R. Kelsey, "Multicast Protocol for Low-Power and Lossy Networks (MPL)," IETF RFC 7731, Feb. 2016.

[17] A. Chaabane, E. De Cristofaro, M.A. Kaafar, E. Uzun, "Privacy in content-oriented networking: Threats and countermeasures," ACM SIGCOMM *Computer Communication Review*, vol. 43, no. 3, 2013.